基于多数覆盖的二级 MPRM 函数逻辑优化

王伦耀*①② 夏银水② 陈偕雄①
①(浙江大学信电系 杭州 310027)
②(宁波大学信息科学与工程学院 宁波 315211)

摘 要：利用不相交乘积项之间逻辑“或”和逻辑“异或”可以互换的特性，该文将原逻辑函数转化成由不相交乘积项组成的二级混合极性 Reed-Muller (MPRM)函数。然后通过搜索不相交乘积项的多数覆盖和检测乘积项间的位操作结果，实现了二级 MPRM 函数的优化。另外，该文还提出一种基于逻辑覆盖的功能验证方法也被提出用于验证逻辑函数优化前后逻辑功能的等效性。实验显示，与已发表的方法相比，该文的优化算法在保证优化效果的同时使运算速度获得了明显的改进。

关键词：数字逻辑电路；Reed-Muller 逻辑；混合极性；逻辑优化；逻辑最小化
中图分类号：TN79+1 **文献标识码**：A **文章编号**：1009-5896(2012)04-0986-06
DOI: 10.3724/SP.J.1146.2011.00915

Two-level MPRM Functions Optimization Based on Majority Cubes

Wang Lun-yao①② Xia Yin-shui② Chen Xie-xiong①
①(*Department of Information Science and Electronic Engineering*, *Zhejiang University*, *Hangzhou* 310027, *China*)
②(*Faculty of Information Science and Engineering*, *Ningbo University*, *Ningbo* 315211, *China*)

Abstract: Based on the property of the disjointed cubes that the logic operators “OR” and “EXOR” can replace each other, an algorithm of two level Mixed-Polarity Reed-Muller (MPRM) optimization is proposed. In the algorithm, by searching and decomposing the majority cubes of these disjointed cubes and replacing them with more compacted and less cubes, a minimized MPRM function is obtained. Further, an efficient approach for logic verification based on logic covers is also presented to check whether two functions are equal or not after logic minimization. The proposed algorithm is implemented in C and tested on MCNC benchmarks. Experimental results show that the proposed method can offer a compacted MPRM expression efficiently in contrast to the reported methods.

Key words: Digital logic circuit; Reed-Muller logic; Mixed polarity; Logic optimization; Logic minimization

1 引言

数字逻辑电路既可以采用基于“与/或/非”运算的布尔(Boolean)逻辑来实现，也可以采用基于“与/异或”运算的 Reed-Muller (RM)逻辑来实现。目前的逻辑综合工具基本上是针对“与/或/非”逻辑，但是利用“与/异或”进行逻辑综合正吸引越来越多研究者的兴趣[1−3]。其原因在于约一半的电路，或者电路的一部分[4,5]，如果利用 RM 逻辑来实现往往可以获得比布尔逻辑更好的综合结果。而且相比于布尔逻辑，RM 逻辑具有良好的可测试性。随着集成电路工艺的发展，目前 RM 逻辑中的“异或”门电路无论面积，速度还是功耗都获得很大的改进，并应用到实际设计中[6]。此外，有关 RM 逻辑在可编程逻辑阵列(FPGA)方面的研究也受到关注[7,8]。

在 RM 逻辑优化中，极性变换是实现 RM 逻辑优化的一个重要方法。RM 逻辑的极性包括固定极性(Fixed Polarity, FP)和混合极性(Mixed Polarity, MP)。在采用固定极性 RM(FPRM)优化中，函数的每个变量只能以原变量或反变量的其中一种形式出现；而在采用混合极性 RM(MPRM)的优化中，变量的取值形式没有限制，因此可以获得更加简化结果。但相比于 FPRM 的 2^n 种极性组合，MPRM 有 3^n 种极性组合，其中 n 为变量个数。因此随着输入变量数 n 的增加，如何快速实现逻辑优化是所有 MPRM 优化算法所面临的挑战。目前 MPRM 优化技术包括真值矢量(truth vector)法[9]，Onset 表(Onset table)法[2]，进化算法[3,10]，列表(tabular technique)法[11]等。

2011-09-05 收到，2011-12-23 改回
国家自然科学基金(60871022, 61041001, 61131001)和宁波市自然科学基金(2101A610183)资助课题
*通信作者：王伦耀 wanglunyao@nbu.edu.cn

除上述方法以外，还有一种方法是采用寻找乘积项的多数覆盖并结合逻辑分离技术进行逻辑优化[12,13]。文献[12]提出了基于最小项的多数覆盖优化技术。采用最小项的好处有：(1)容易将被覆盖的最小项从原函数中分离出来；(2)容易计算新生成项。但不足之处在于最小项的个数与输入变量数成指数级关系，这使得上述算法在处理大电路时速度大受影响，甚至失效。因此，文献[12]仅提出了逻辑优化设想而并没有将该设想用于实际电路的 MPRM 优化。文献[13]利用多数覆盖技术并结合 b_j 图[14]提出了 MPRM 逻辑优化方法。该方法由两个步骤组成：(1)将待优化的布尔逻辑转化为最佳极性(best-polarity)下的二级 FPRM 逻辑；(2)再利用多数覆盖和 b_j 图将最佳极性的二级 FPRM 逻辑转化为二级 MPRM 逻辑。但由于最佳极性搜索是一个很耗时的过程，从而使文献[13]方法的效率受到影响。

本文将采用基于不相交乘积项的多数覆盖技术进行二级混合极性 RM 逻辑优化。相比于文献[12]中的最小项，逻辑函数的不相交乘积项数量往往远小于最小项数量，因此可以有效减少算法运算量，从而提高算法效率。另外，由于不相交乘积项之间逻辑“或”和“异或”可以互换，这给布尔逻辑向 MPRM 逻辑转化带来方便；同时也不需要文献[13]中的极性搜索这个步骤，有助于算法效率的提高。此外，在逻辑等效验证方面，本文提出了基于逻辑覆盖的逻辑等效验证方法。该方法通过比较逻辑函数优化前后所对应的逻辑覆盖是否相等的方法来验证逻辑函数综合前后是否等效。相比于最小项的验证方法，本方法能有效避免在大逻辑函数验证中因最小项数量增加而导致验证效率低下的问题。

2　定义

对于一个 n 变量逻辑函数 f 总可以写成式(1)的形式。

$$f=\sum_{i=1}^{k}p_i \tag{1}$$

式(1)中，k 为乘积项的个数；“Σ”表示乘积项之间是逻辑“或”关系；p_i 为乘积项，$p_i=x_1x_2\cdots x_j\cdots x_n$，$x_j\in\{0,1,-\}$，分别表示 x_j 取反变量，原变量和该变量不出现。

对于任何一个如式(1)所示逻辑函数也可以转化成具有式(2)形式的 RM 逻辑函数 g。

$$g=\oplus\sum_{i=1}^{m}q_i \tag{2}$$

其中“$\oplus\Sigma$”表示构成 g 的各个乘积项之间是逻辑“异或”关系。式(2)中，q_i 为乘积项，m 为乘积项的个数。如果函数 g 包含 n 个变量，则乘积项 q_i 也可以写成 p_i 的形式。

定义 1　乘积项 p_w,p_v 的 0-1 位“异或”操作，记为 $\mathrm{HD}(p_w,p_v)_{01}$。设 x_{wi} 和 x_{vi} 分别为乘积项 p_w,p_v 的第 i 位变量，$x_{wi},x_{vi}\in\{0,1,-\}$；并规定 $0\widehat{_{01}}0=0$，$0\widehat{_{01}}1=1$，$1\widehat{_{01}}1=0$，$1\widehat{_{01}}-=/$，$0\widehat{_{01}}-=/$，$-\widehat{_{01}}-=/$。其中“$\widehat{_{01}}$”表示 p_w,p_v 对应位取值为 0 或 1 的变量进行“异或”操作；“/”表示没有意义。$\mathrm{HD}(p_w,p_v)_{01}$ 的值等于 p_w 和 p_v 进行 $\widehat{_{01}}$ 位操作后生成的“1”的个数。

定义 2　乘积项 p_w,p_v 的 dc 位“异或”操作，记为 $\mathrm{HD}(p_w,p_v)_{dc}$。设 x_{wi} 和 x_{vi} 分别为乘积项 p_w,p_v 的第 i 位变量，$x_{wi},x_{vi}\in\{0,1,-\}$；并规定 $0\widehat{_{dc}}0=/$，$0\widehat{_{dc}}1=/$，$1\widehat{_{dc}}1=/$，$1\widehat{_{dc}}-=1$，$0\widehat{_{dc}}-=1$，$-\widehat{_{dc}}-=0$。其中“$\widehat{_{dc}}$”表示 p_w,p_v 对应位取值至少一个为“-”的变量进行位“异或”操作；“/”表示没有意义。$\mathrm{HD}(p_w,p_v)_{dc}$ 的值等于 p_w 和 p_v 进行 $\widehat{_{dc}}$ 位操作后生成的“1”的个数。

图 1 分别显示了两个乘积项的 0-1 和 dc 位“异或”操作的结果。从定义 1 和定义 2 可以看出，当 $\mathrm{HD}(p_w,p_v)_{dc}=0$ 时，$\mathrm{HD}(p_w,p_v)_{01}$ 就是传统意义上的海明距。

	1	-	0	1: p_1
$\widehat{_{01}}$	0	-	-	1: p_2
	1	/	/	0

(a)乘积项 0-1 位“异或”操作

	1	-	0	1: p_1
$\widehat{_{dc}}$	1	-	-	1: p_3
	/	0	1	/

(b)乘积项 dc 位“异或”操作

图 1　乘积项的位“异或”操作

3　二级 MPRM 优化算法

3.1　基于多数覆盖优化二级 MPRM 优化算法

如果两个乘积项 p_w 和 p_v 不相交，即 $p_w\cdot p_v=0$，则可以得到式(3)。

$$p_w+p_v=p_w\oplus p_v+p_wp_v=p_w\oplus p_v \tag{3}$$

因此，当构成 f 的各个乘积项均为不相交乘积项时，则式(1)可以转化为式(4)。

$$f=\sum_{i=1}^{k}p_i=\oplus\sum_{i=1}^{k}p_i \tag{4}$$

在式(4)中，由于没有对乘积项中各个变量的取值形式进行限制，即对于某个变量 x_i 可以以原变量和反变量的形式同时出现在 f 中，所以，式(4)对应的是二级 MPRM 表达式。将逻辑函数乘积项转换为不相交乘积项的集合可以利用乘积项之间的不相交锐积实现[5]。

对于一个 n 变量逻辑函数 f，令 C_{on}，C_{off}，C_{dc}

分别表示 f 取逻辑“1”的乘积项集合(简称 ONSET),取逻辑“0”的乘积项集合(简称 OFFSET)和不顾项集合(简称 DCSET)；并令 C_{on} 的对应的不相交乘积项集合为 C_{fds}，则 f 的全集 C 可以表示为式(5)。

$$C = C_{on} \bigcup C_{off} \bigcup C_{dc} = C_{fds} \bigcup C_{off} \bigcup C_{dc} \tag{5}$$

令 C^m 为 C 中相邻 2^m 个最小项组成的一个集合；C_{poff} 为 C^m 中属于 OFFSET 的乘积项集合，即 $C_{poff} = C^m \bigcap C_{off}$；$C_{pon}$ 为属于 ONSET 或 DCSET 并且与 C^m 相交或被 C^m 包含的乘积项集合。即对任一乘积项 p_i，若 $p_i \in C_{pon}$，则 $\{p_i\} \bigcap C^m \neq \varnothing$。$C_{pon}$ 可以表示为式(6)。

$$C_{pon} = C^m \oplus C' \oplus C_{poff} \tag{6}$$

其中 $C' \subseteq C_{pon}, C' \bigcap C^m = \varnothing$。

令 $C_{pon} \to \sum_{i=1}^{k} p_i$，$C_{poff} \to \sum_{j=1}^{r} p_j$，$C' \to \sum_{l=1}^{z} p_l$，$C^m \to p^{n-m}$，则式(6)可以表示为

$$\sum_{i=1}^{k} p_i = P^{n-m} \oplus \sum_{l=1}^{z} p_l \oplus \sum_{j=1}^{r} p_j = P^{n-m} \oplus \sum_{s=1}^{w} p_s \tag{7}$$

因此，当 $k > w+1$ 时，其中 $w = z + r$，表达式 $\left(P^{n-m} \oplus \sum_{s=1}^{w} p_s\right)$ 中的乘积项数要比 $\sum_{i=1}^{k} p_i$ 的少，从而实现逻辑的简化。以往的研究表明[12]，当 C^m 包含的 2^m 个最小项中至少 3/4 属于 $C_{on} \bigcup C_{dc}$ 时，采用式(7)方式来表示原函数往往可以获得更加简化的形式。C^m 也被称为多数覆盖。式(7)实际上提供了一种简化逻辑函数的方法。这个方法包含两个主要步骤：(1)寻找多数覆盖 C^m；(2)分离出乘积项集合 $\sum_{s=1}^{w} p_s$。

在本文中，基于不相交乘积项的多数覆盖搜索可以通过图 2 所示 onset 表来实现。图 2(a)是 f 的卡诺图，图 2(b)是函数 f 的 onset 表[2]。该表分为 3 部分，自上而下分别记为 T_{up}，T_m 和 T_l。其中 T_{up} 列出了 f 的不相交乘积项。T_m 是对 T_{up} 中各个变量取值形式的统计，并且自上而下依次记录取反变量、原变量和变量不出现的情况。例如在图 2(b)所示的 T_m 中，第 0 行为(2131)，表示变量 (a,b,c,d) 以反变量形式出现在 T_{up} 的次数分别为 2 次，1 次，3 次和 1 次。T_l 表示各变量的取值情况，“0”对应反变量，“1”对应原变量，“-1”和“2”表示对应变量不出现。

借助图 2(b)的 onset 表，基于不相交乘积项的多数覆盖 MPRM 算法的主要步骤描述如下：

(1)由 f 对应的不相交乘积项构成图 2(b)所示的 T_{up}，并将 T_l 各位初始化为“-1”；

(2)统计 T_{up} 中各个变量取值情况，并将结果寄存到 T_m；

(3)检测 T_m 第 2 行(即最后一行)中取值最大的

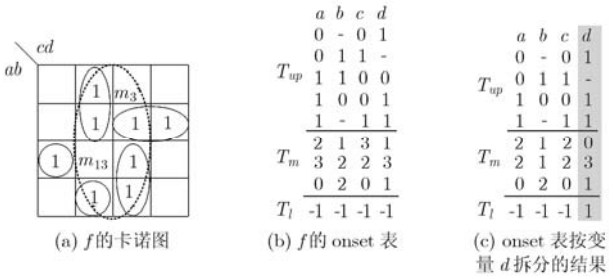

图 2 逻辑函数的 onset 表表示及 onset 表拆分

数是不是等于 T_{up} 的行数；如果是，确定该最大数在 T_m 中的位置，设为 $(2, L_2)$，则令 $T_l(L_2) = 2$；

(4)搜索 T_m 中前 2 行(第 0 行和第 1 行)中最大数所在的位置，记为 (R, L)；如果存在 k 个相同的最大数 $(k > 1)$，记录这些数的位置 (R_i, L_i)，$i = (0,1,\cdots, k-1)$，并进一步比较在 $(2, L_i)$ 上数据大小，然后选定最大的一个数，记录它的位置；如果在 $(2, L_i)$ 上有多个数据相等，则任选其中一个，并记录它的位置为 $(2, L_i)$，令 $T_l(L_i) = R_i$；

(5)将 T_{up} 拆分成两部分。将 T_{up} 中第 L_i 列变量取值与 $T_l(L_i)$ 相等或者为“-”的乘积项提取出构成新的 T_{up}，而将剩下的乘积项构成 T'_{up}，并删除原来的 T_{up}；

(6)检测新的 T_{up} 是否符合多数覆盖约束，若符合约束，执行步骤(7)；否则执行步骤(8)；

(7)删除 T_{up}，并生成与式(7)对应的 p^{n-m}，$\oplus\sum_{l=1}^{z} p_l$ 和 $\oplus\sum_{j=1}^{r} p_j$；将 $\oplus\sum_{l=1}^{z} p_l$ 和 $\oplus\sum_{j=1}^{r} p_j$ 添加到 T'_{up}，并令更新后的 T'_{up} 为 T_{up}；

(8)重复执行步骤(2)到步骤(6)，直到找不到符合多数覆盖约束的 T_{up} 为止。

在步骤(6)中判断 T_{up} 是否符合多数覆盖约束的方法如下：

(a)统计 T_l 中“-1”和“2”的数量，分别记为 A_1 和 A_2，并令 $m = A_1 + A_2$。

(b)在 T_{up} 中确定 T_l 中“-1”所对应的列，并统计这些列中“-”的数量，记为 A_3；同时统计这些列中没有“-”的乘积项个数，记为 A_0；

(c)令 $N = (2^{A_3} + A_0) \times 2^{A_2}$。当 $N \geq (3/4) \times 2^m$，表示符合多数覆盖约束，否则不符合。

在步骤(7)中乘积项组合 $\oplus\sum_{l=1}^{z} p_l$ 和 $\oplus\sum_{j=1}^{r} p_j$ 生成方法如下：假设 $\oplus\sum_{j=1}^{r} p_j \to C_{poff}$，$\oplus\sum_{l=1}^{z} p_l \to C'$，则 $C_{poff} = P^{n-m} \otimes T_{up}$，$C' = T_{up} \otimes P^{n-m}$。符号“$\otimes$”表示乘积项不相交锐积。$P^{n-m}$ 的表达式由 T_l 转化而来，就是将 T_l 中“1”所在的列对应的变量以原变量形式出现；“0”以反变量形式出现；“-1”

或“2”则表示对应的变量不出现。如图 2(c)中的T_l对应的P^{n-m}为$P^{n-m}=d$。由于$C_{poff}=P^{n-m}\otimes T_{up}$，$C'=T_{up}\otimes P^{n-m}$，因此，构成$C_{poff}$的乘积项不属于$T_{up}$且不相交；而构成$C'$的乘积项属于$T_{up}$且不相交，所以属于$C_{poff}\cup C'$的乘积项都互不相交。因此，更新后的$T_{up}$所包含的乘积项也不相交。以图 2(a)为例，$C_{poff}=\{m_3,m_{13}\}=\{0011,1101\}$与$C'=\{0110\}$不相交。

现以图 2(b)为例说明多数覆盖的搜索和覆盖拆分过程。图 2(b)所示的T_m中最大的数为 3，且有 3 个，它们的位置分别为(1,0), (0,2), (1,3)。因此需检测位于(2,0), (2,2), (2,3)上的数据大小，显然在(2,3)位置上的数据取值最大，即$T_l(3)=1$。检测图 2(b)T_{up}的第 3 列，将取值不为“1”或“-”的乘积项移除，得到图 2(c)所示的T_{up}。接下来进行多数覆盖约束判断。以图 2(c)为例，由T_l得到$A_1=3$，$A_2=0$，$2^m=2^{A_1+A_2}=8$。另外，在T_{up}中，在变量$\{a,b,c\}$所在的列中，有$A_3=2$，$A_0=2$，所以$N=(2^{A_3}+A_0)\times 2^{A_2}=6$，满足$N\ge(3/4)\times 2^m$。对照图 2(a)不难看出，虚线部分对应的就是乘积项$P^{n-m}=d$，它包含了 8 个最小项，其中 6 个取值为“1”。

3.2 基于乘积项位操作的二级 MPRM 优化方法

逻辑函数经过多数覆盖二级 MPRM 优化以后，有些乘积项可以通过式(8)，式(9)，式(10)所示的方法进行进一步的简化。

$$\left.\begin{aligned} a\bar{b}\oplus\bar{a}b=a\oplus b\\ \bar{a}\bar{b}\oplus ab=\bar{a}\oplus b\end{aligned}\right\} \tag{8}$$

$$\bar{a}b\oplus ab=b \tag{9}$$

$$a\oplus ab=a\bar{b} \tag{10}$$

其中利用式(8)可以减少表达式中的字母数，式(9)和式(10)可以减少乘积项数。上述简化可以利用乘积项的位操作实现。假设经过多数覆盖二级 MPRM 优化得到的乘积项集合为C_{RM}，则基于位操作优化算法的主要步骤如下：

(1)在C_{RM}中任取两个乘积项p_w和p_v，计算它们的位操作结果$\mathrm{HD}(p_w,p_v)_{01}$和$\mathrm{HD}(p_w,p_v)_{dc}$；

(2)若满足$\mathrm{HD}(p_w,p_v)_{01}=2$，$\mathrm{HD}(p_w,p_v)_{dc}=0$，则用式(8)化简；若满足$\mathrm{HD}(p_w,p_v)_{01}=1$，$\mathrm{HD}(p_w,p_v)_{dc}=0$，则用式(9)化简；若满足$\mathrm{HD}(p_w,p_v)_{01}=0$，$\mathrm{HD}(p_w,p_v)_{dc}=1$，则用式(10)化简；删除$p_w$和$p_v$，同时将新生成的乘积项放入$C_{\mathrm{RM}}$中；

(3)循环执行步骤(1)，步骤(2)，直到没有乘积项组合满足步骤(2)规定的位操作结果为止。

以图 2(a)所示的逻辑函数为例。经过基于多数覆盖二级 MPRM 优化后，得到乘积项集合为：$C_{RM}=\{---1,1100,1101,0011,0110\}$。在经过乘积项位操作优化算法后，$C_{\mathrm{RM}}$最后被简化为$\{---1,110-,001-,0-10\}$。

3.3 逻辑功能验证

逻辑功能验证是逻辑综合中重要的一环。逻辑函数f经优化后得到了另外一个函数f'，为了检验f和f'逻辑功能是否一样，必须进行逻辑功能验证(functional verification)。一种简单的验证方法就是检查f和f'对应的最小项是不是一样。但是这种方法由于最小项的个数与输入变量数呈指数级增加，因此在大电路验证上，基于最小项的验证方法的效率受到严重影响。在本文中，将采用覆盖比较的方法来实现逻辑函数优化前后的功能验证。

令C_{on}，C_{off}和C_{dc}分别表示逻辑函数f的 ONSET, OFFSET 和 DCSET；C'_{on}，C'_{off}和C'_{dc}分别表示逻辑函数f'的 ONSET，OFFSET 和 DCSET。显然，若$f=f'$，则有：$C_{on}=C'_{on}$，$C_{off}=C'_{off}$。但考虑到在优化过程中，部分属于C_{dc}的乘积项会被吸收到C'_{on}中，所以f和f'是否功能等效可以通过检测下面两个条件是否同时被满足来判断。

条件 1：$C_{on}\subseteq C'_{on}$；

条件 2：$C'_{on}\subseteq C_{on}\cup C_{dc}$。

条件 1 表示f的 ONSET 被f'的 ONSET 所覆盖；条件 2 表示f'的 ONSET 被f的 ONSET 和 DCSET 覆盖，亦即f'的 ONSET 与f的 OFFSET 没有交集。

在本文中，由于f'是 RM 逻辑函数，考虑到$0\oplus 0=0$，$1\oplus 1=0$，因此部分原属于 OFFSET 的乘积项也会被包含在f'的乘积项集合C_{RM}中，如图 2(a)中的m_{13},m_3，从而导致$C_{\mathrm{RM}}\ne C'_{on}$。因此在逻辑功能验证过程中必须先除去$C_{\mathrm{RM}}$中属于OFFSET的那部分覆盖，得到对应的$C'_{on}$。为此本文引入一个新的算子，称为双锐积算子，用符号“◊”表示。它定义为

$$p_w\Diamond p_v=(p_w\otimes p_v)\cup(p_v\otimes p_w) \tag{11}$$

考虑到$p_w\otimes p_v=p_w\bar{p}_v$，$p_v\otimes p_w=p_v\bar{p}_w$，因此$p_w\Diamond p_v$可以表示为

$$p_w\Diamond p_v=p_w\bar{p}_v+\bar{p}_wp_v=p_w\oplus p_v \tag{12}$$

所以$p_w\Diamond p_v$对应的覆盖就是$p_w\oplus p_v$对应的 ONSET 覆盖。此外，对不相交锐积而言，当$C_1\subseteq C_2$时，存在$C_1\otimes C_2=\varnothing$。因此可以通过判断$C_{on}\otimes C'_{on}$和$C'_{on}\otimes(C_{on}\cup C_{dc})$是否同时为空集来检测上述条件 1 和条件 2 是否同时满足，进而判断f和f'是否功能等效。

逻辑功能验证主要步骤描述如下：

(1)在C_{RM}中任取两个乘积项p_w和p_v，如果p_w

和 p_v 相交，执行 $p_w \Diamond p_v$，新生成的乘积项添加到 C_{RM}，同时删除 p_w 和 p_v；

(2)循环执行步骤(1)，直到 C_{RM} 中任何两个乘积项都不相交，此时 C_{RM} 已转化为 $C_{on}^{'}$；

(3)在 C_{on} 中任取出一个乘积项 p_w，并令 $C_v = p_w \otimes C_{on}^{'}$，若存在 $C_v \neq \varnothing$，则表明条件 1 不满足，逻辑函数不等效，结束运算。若 C_{on} 中所有乘积项都满足 $C_v = \varnothing$，则表明条件 1 满足，执行步骤(4)；

(4)在 $C_{on}^{'}$ 中任取出一个乘积项 p_v，并令 $C_v = p_v \otimes (C_{on} \bigcup C_{dc})$，若存在 $C_v \neq \varnothing$，则表明条件 2 不满足，逻辑函数不等效，结束运算。若 $C_{on}^{'}$ 中所有乘积项都满足 $C_v = \varnothing$，则表明条件 2 满足，逻辑函数等效。

4 实验及结果分析

本文的二级 MPRM 逻辑优化算法用 C 语言实现，并在操作系统为 Windows XP, CPU 时钟频率为 2.1 GHz，内存为 2 GB 的 PC 机上实现。表 1 是文献[10]的结果与本文方法的比较情况。表中测试电路来自 MCNC Benchmark。其中“in/out/p”表示测试电路的输入、输出和原始乘积项数。P_{GA} 表示文献[10]用遗传算法(GA)得到的二级 MPRM 的乘积项数。P_{mj} 表示采用本文方法得到的二级 MPRM 的乘积项个数。T_{GA} 为文献[10]的方法在 Windows 操作系统，CPU 时钟频率为 2.4 GHz，内存为 2 GB 的 PC 机上的运行时间，计时单位是“s”。T 是本文方法的运行时间，计时单位是“ms”。

考虑到数字电路的面积与逻辑函数的乘积项数量密切相关，因此可以通过比较逻辑函数乘积项的个数来间接衡量电路的面积大小。从表 1 可得，与文献[10]的方法相比，超过一半的电路在使用本文的方法后在面积上可以得到进一步简化。在运行时间上，表 1 中，“<1”表示算法显示时间为“0 ms”。文献[10]算法运行环境和本文方法运行的环境很接近。但两种方法占用的 CPU 时间差异很大。这点从文献[10]中采用的计时单位以及那些运行时间显示不为“0”的电路就可以看出来。

表 1 本文方法与文献[10]方法的比较结果

电路	in/out/p	P_{GA}/P_{mj}	T_{GA}(s)	T (ms)
Rd73	7/3/141	63/96	~0	<1
Rd84	8/4/256	107/214	~0	<1
Misex1	8/7/32	13/31	~0	<1
Sao2	10/4/58	76/92	~0	<1
5xp1	7/10/75	61/67	~0	<1
Misex3	14/14/1848	1421/1732	10	500
Rd53	5/3/32	20/19	~0	<1
Con1	7/2/9	14/8	~0	<1
Clip	9/5/167	182/156	~0	15
Ex1010	10/10/1024	810/390	~0	93
Inc	7/9/35	34/33	~0	<1
Table3	14/14/175	401/182	6	<1
Table5	17/15/159	551/170	78	<1
Alu4	14/8/1028	2438/1265	7	344

表 2 给出了本方法与文献[13]方法的比较结果。其中 P_1 表示文献[13]的方法得到的乘积项个数，P_{mj} 为本文的方法得到的乘积项个数。文献[13]的方法采用了两个步骤实现二级 MPRM 逻辑优化。其中第 1 步将测试电路转化为二级最佳极性(best polarity)下的 RM 表达式；在第 2 步中将最佳极性下的 RM 表达式转换为二级 MPRM 表达式。表 2 中“$T_1/T_2/T_3$”分别表示文献[13]的算法在第 1 步、第 2 步和总的运行时间。从表 2 中可以看出，文献[13]的最佳极性搜索时间 T_1 占总时间的绝大部分；而且 T_1 大小与输入变量的数量基本上成正相关。相比之下，本文的方法的运算速度与被优化电路的乘积项数量有关，而对电路的输入变量数量不敏感。如在表 2 中，虽然有些电路输入变量超过 20 个，但由于乘积项数量较少，如 Cm150a, Mux 等，它们运算时间都没有超过 1 ms。而在表 1 中，虽然有的测试电路输入的变量数不到 20，但乘积项的数量比较大，如 Alu4, Misex3 等，使得运算时间最高达到 500 ms。本方法的这种特性在处理那些输入变量比较多而乘积项较少的电路具有很高的效率。在优化后乘积项的数量上，本文方法和文献[13]的方法比较接近。

表 2 本文方法与文献[13]方法的比较结果

电路	in/out/p	P_1/P_{mj}	$T_1/T_2/T_3$ (ms) *	T (ms)
5xp1	7/10/75	63/67	60/30/90	<1
Cmb	16/4/54	5/6	17650/10/17660	<1
Rd84	8/4/256	107/214	160/30/190	<1
Rd73	7/3/141	63/96	80/20/1000	<1
Pm1	16/13/37	25/32	34180/30/34210	<1
Cm150a	21/1/17	17/17	512350/20/512370	<1
Mux	21/1/36	16/16	287760/10/287770	<1
F51m	8/8/76	51/48	120/10/130	<1
Pcle	19/9/45	26/21	263000/30/263030	<1
Rd53	5/3/32	20/19	50/10/60	<1
Tcon	17/16/32	25/24	56220/40/56260	<1

*PC 机配置： Linux 操作系统， Intel Pentium-266 CPU, 64 MB 内存。

5 结论

本文提出了一种新的二级 MPRM 函数逻辑优化算法。该算法大体包含下面两个部分：(1)不相交乘积项多数覆盖优化算法；(2)乘积项位操作方法。由于本文先将原逻辑函数的乘积项转化为不相交乘积项，由于一般情况下，不相交乘积项的个数远小于最小项的个数，这使得本算法需处理的数据量大为减少，从而使本算法具有更快的运算速度。从实验结果来看，本方法在保证逻辑优化效果情况下，优化的速度获得了显著的改进。另外，本文算法的运算速度具有对逻辑函数输入变量个数不敏感的特点。这使得本文算法在处理大电路，尤其是处理输入变量比较多，乘积项个数比较少的电路时效率往往比较高。

此外，本文也提出了基于逻辑覆盖的逻辑功能验证算法，与基于最小项的逻辑功能验证算法相比，本文提出的验证算法可以有效避免在处理大电路时，因输入变量数量增加而引起最小项数量呈指数级增加，导致验证算法效率低下甚至无法运算的情况。

参 考 文 献

[1] Pradhan S N and Chattopadhyay S. Two-level AND-XOR networks synthesis with area-power trade-off[J]. *International Journal of Computer Science and Network Security*, 2008, 8(9): 365–375.

[2] Wang L, Xia Y, Chen X, *et al*.. Reed-Muller function optimization techniques with onset table [J]. *Journal of Zhejiang University Science-C*, 2011, 12(4): 288–296.

[3] Yang M, Xu H, and Almaini A E A. Optimization of mixed polarity Reed-Muller functions using genetic algorithm [C]. The 3rd International Conference on Computer Research and Development, Shanghai, China, 2011, Vol.3: 293–296.

[4] Xia Y, Sun F, and Mao K. A detection method for logic functions suitable for dual-logic synthesis [J]. *Progress in Natural Science*, 2009, 19(10): 1311–1315.

[5] Wang L, Xia Y, and Chen X. Detection and decomposition algorithm for dual logic implementation [C]. The 12th IEEE International Confrence on Communication Technology, Nanjing, China, 2010: 1426–1429.

[6] Navi K, Maeen M, Foroutan V, *et al*.. A novel low-power full-adder cell for low voltage [J]. *The VLSI Journal of Integration*, 2009, 42(4): 457–467.

[7] Muma K, Chen D, Choi Y, *et al*.. Combining ESOP minimization with BDD-based decomposition for improved FPGA synthesis [J]. *Canadian Journal of Electrical and Computer Engineering*, 2008, 33(3): 77–182.

[8] Seok-Bum K. Area minimization of exclusive-OR intensive circuits in FPGAs [J]. *Journal of Electronic Testing: Theory and Applications*, 2004, 14(3): 219–225.

[9] Xia Y, Wang L, Zhou Z, *et al*.. Novel synthesis and optimization of multi-level mixed polarity Reed-Muller functions [J]. *Journal of Computer Science and Technology*, 2005, 20(6): 895–900.

[10] Al Jassani B A, Urquhart N, and Almaini A E A. Manipulation and optimization techniques for Boolean logic [J]. *IET Computers & Digital Techniques*, 2010, 4(3): 227–239.

[11] 李辉, 汪鹏君, 王振海. 混合极性列表技术及其在 MPRM 电路面积优化中的应用[J]. 计算机辅助设计与图形学学报, 2011, 23(3): 527–533.
Li Hui, Wang Peng-jun, and Wang Zhen-hai. Tabular techniques for mixed polarity and its application in area optimization of MPRM circuits [J]. *Journal of Computer Aided Design & Computer Graphics*, 2011, 23(3): 527–533.

[12] Tran A and Wang J. Decomposition method for minimization of Reed-Muller polynomials in mixed polarity [J]. *IEE Proceedings E Computers and Digital Techniques*, 1993, 140(1): 65–68.

[13] Wang L and Almaini A E A. Optimization of Reed-Muller PLA implementations [J]. *IEE Proceedings Circuits Devices & Systems*, 2002, 149(2): 119–128.

[14] Wu X, Chen X, and Hurst S L. Mapping of Reed-Muller coefficients and minimization of exclusive-OR switching functions [J]. *IEE Proceedings E Computers and Digital Techniques*, 1982, 129(1): 15–20.

王伦耀： 男，1972 年生，博士生，副教授，研究方向为低功耗数字电路设计、数字集成电路逻辑综合和优化、高信息密度集成电路设计.

夏银水： 男，1963 年生，博士，研究员，博士生导师，研究方向为低功耗数字电路设计、数字集成电路逻辑综合和优化以及 SoC 设计.

陈偕雄： 男，1941 年生，男，教授，博士生导师，研究方向为近代数字电子学、高信息密度集成电路设计、谱技术及其应用等.